AD-A171 159

THE ISIS PROJECT: Quarterly R&D Status and Technical Report

May. 4, 1986 - Aug 1, 1986

Kenneth P. Birman

DTIC
ELECTE
AUG 2 2 1986
S D
D

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

DTIC FILE COPY

This work was sponsored by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 5378, under contract MDA903-85-C-0124 issued by the department of the Army.

The view, opinions and findings contained in this report are those of the authors and should not be construed as an official DoD position, policy, or decision.

REPORT DOCUMENTATION PAGE

AD-A171159

OMB No 0704-0188
Exp Date Jun 30, 1986

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) 407 072	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Kenneth P. Birman, Assist. Prof CS Dept., Cornell University 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Defense Advanced Research Projects Agency/IPTO
6c ADDRESS (City, State, and ZIP Code) Computer Science Department 405 Upson Hall Cornell University, Ithaca, NY 14853	7b ADDRESS (City, State, and ZIP Code) Defense Advanced Research, Project Agency Attn: TIO/Admin, 1400 Wilson Blvd. Arlington, VA 22209
8a NAME OF FUNDING/SPONSORING ORGANIZATION DARPA/IPTO 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER ARPA order 5378 Contract MDA903-85-C-0124
8c ADDRESS (City, State, and ZIP Code) Defense Advanced Research, Project Agency Attn: TIO/Admin., 1400 Wilson Blvd. Arlington, VA 22209	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

Quarterly R & D Status Report

Approved for Public Release
Distribution Unlimited

12 PERSONAL AUTHOR(S)
Kenneth P. Birman

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Quart. R&D Status Rep.	FROM 5/4/86 TO 8/4/86	8/4/86	7

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This quarterly R & D Status report covers the period between May 4, 1986 and August 4, 1986.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	
22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code) / 22c OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Academic Staff

Kenneth P. Birman, Principal Investigator

Thomas A. Joseph, Research Associate

Graduate Students

Amr. El. Abbadi

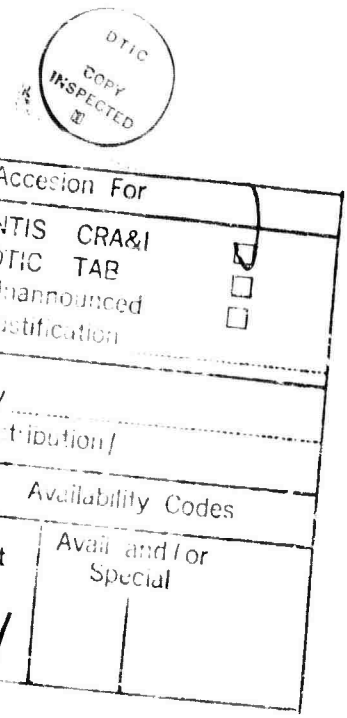
Kenneth Kane

Richard Koo

Frank Schmuck

Patrick Stephenson

Joseph Touch



1. Description of Progress

This report summarizes accomplishments of the *ISIS* project during the period May 4, 1986 - Aug 4, 1986. We assume that the reader is familiar with the goals of the project and has read some of our recent progress reports. Accordingly, the summary will be brief and targeted to specific accomplishments made during this period, rather than the overall status of the project.

During the second quarter of 1986, the *ISIS* effort focused on the continued development of a new system for supporting fault-tolerant process groups and software built using such groups. We have now begun to see widespread acceptance of this work among other researchers who have struggled with synchronization issues in fault-tolerant distributed computing. By creating an environment within which fault-tolerant distributed software can be constructed with very little attention to sychronization, our approach makes it possible to solve problems that could take months using conventional programming methods in days. Meanwhile, we have fleshed out our own objectives, which are now to construct a complete support mechanism, imbedded into a distributed operating system, for simplified distributed and fault-tolerant computing.

ISIS is certainly not the only project working in this problem domain, but we feel that it has been one of the most successful. The reason is essentially that where other groups have based their effort on "blindingly fast remote procedure calls" (RPC) or "high speed multicast protocols", *ISIS* has focused on integration of tools such as these into a complete, coherent environment. When doing so, we found that some of the basic mechanisms needed simply can't be expressed in terms of RPC or conventional multicast protocols, and came up with our fault-tolerant process group approach to address them. Thus, where other efforts have generally concluded that it is extremely hard to build fault-tolerant programs, *ISIS* has had repeated successes in this domain. Our current work essentially seeks to package the mechanisms on which *ISIS* is based in ways that will make them as accessible as possible to non-expert programmers. These packaged solutions come in several flavors: *fault-tolerant process groups*, which combine a reliable multicast RPC mechanism with a failure handling mechanism, *resilient objects*, which are pro-

grammed using a specification language in which failures and replication are not explicitly addressed, and then translated automatically into a fault-tolerant distributed program having the desired behavior, and *bulletin boards*, a new approach to support interactions between asynchronous programs in fault-tolerant distributed settings. Our premise is that if a wide variety of tools can be provided to the distributed systems programmer, and if these tools are designed to overcome many of the difficulties associated with execution in the presence of failures, then fault-tolerant distributed software will eventually be as easily developed as conventional software. And, we think that to date our results bear out this approach -- a claim few others could make.

1.1. Fault-tolerant process groups

The crux of our present effort is to develop system support for *fault-tolerant process groups* [1]. Such a group consists of a set of processes that cooperate to implement some fault-tolerant distributed service. Unlike other process group approaches, ours is well behaved despite concurrency, failures, recoveries, and dynamic reconfiguration. Moreover, members of a fault tolerant process group can deduce the sequence of events of that *other* members have experienced without running a special protocol for this purpose. In conventional approaches [2] [3], it is hard to determine what other members have observed, hence specialized agreement protocols tend to proliferate throughout any application programs that are expected to perform reliably despite failures, making it very difficult to debug such programs or to be at all confident that they are correct. Our approach essentially pushes such protocols to a lower level of the communication system, where client programs need not be aware of them. It also achieves high levels of concurrency and makes it easy to monitor other processes for failure. Thus, high level code is simpler to develop and debug, and good performance can still be achieved.

When we started working on this approach we showed how it can be applied to clusters of workstations, as might be seen in a typical local area network. During the past few weeks, we succeeded in extending the protocols to make them useful in wide area networks too, completely transparently. This extended treatment is described in a technical report [4], and will be published

in the ACM *Transactions on Computer Systems* shortly. Meanwhile, our implementation of the protocols is advancing smoothly, and we hope to have them running by early fall.

1.2. Resilient objects

Resilient objects are basically distributed programs that mimic the behavior of an abstract data type. They are useful in distributed database applications, and are interesting because we have shown how to compile them automatically from specifications that doen't talk about fault-tolerance at all. The output of this compilation is a fault-tolerant process group. Thus, the programmer who works with resilient objects need not be involved in the details of fault-tolerance. Our new system will support resilient objects, but unlike in the prototype (where such objects were the only facility supported), they will now be integrated with the other mechanisms described here.

Our work on resilient objects has wound down over the past year, and we are now beginning to tie up the last loose ends. Ex-graduate student T. Raeuchle recently defended a Ph.D. dissertation examining concurrency control issues in this area [6]; he shows how to take advantage of *semantic* information to obtain very low cost concurrency control mechanisms for a wide variety of objects. Another graduate student, W. Deitrich, has worked on dynamic data migration in such objects, and is nearly finished with an implementation that will demonstrate his methods. Deitrich is hoping to write his dissertation this fall and defend it this coming winter.

1.3. Fault-tolerant bulletin boards

A fault-tolerant bulletin board is an asynchronous shared memory mechanism that provides guarentees of consistency despite failures and concurrency. Previous work on communication mechanisms of this sort did not address consistency issues and behavior after failures, hence it was hard to talk about fault-tolerance when using this approach to distributed computing. On the other hand, several projects have shown that the approach is suprisingly powerful if failures and consistency are not the primary concern. Our work extends this foundation to address both of

these issues as well. We described this work in our earlier progress reports, pointing to a forthcoming technical report [5]. That report has now been completed, and will be available within the next week or two. In addition to describing the approach, it gives simple examples for a variety of classic problems, like detecting deadlock or performing a transaction on a replicated database, which are hard to solve using standard techniques. Support for this approach is expected to be an important part of our new *ISIS* system.

1.4. Other areas of activity

We are continuing our work on parallel versions of our distributed software development techniques, and hope to complete a paper on this topic during the next quarter of 1986. Work is also continuing in the areas of partitioning, real time control, and command languages for process control.

2. Travel

Birman visited the ANSA group in Cambridge, England, where he was invited to speak to a workshop developing networking standards for use by industry in interconnecting products. However, all travel expenses were paid by the ANSA group.

3. Budget summary

We conclude with a summary of the financial status of the project, which is close to projections in all categories. Notice that some funds have been shifted from the student support line into a hardware line. This was for the purchase of memory boards to upgrade our SUN 2/50 workstations to have 5 Mbytes of memory each, and was done with the permission of Program Director Dennis Perry.

Expenditures - 5/5/86 - 8/4/86

	Planned budget for period	Expenses for period	Prior Expenses	Total to 8/4/86
Secretary support	1,542	1,542	1,028	2,570
Summer faculty	8,956	8,956	20,609	29,565
Research Associate	7,518	7,518	15,144	22,662
Programmer	9,000	9,000		9,000
Graduate students	4,800	4,800	96,039	100,839
Employee benefits	6,218	6,218	6,670	12,888
Computer maintenance			5,025	5,025
Publications	327		2,374	2,374
Supplies	253	249	3,553	3,802
Computer Supplies			643	643
Travel			13,103	13,103
Programmer	1,006	1,006		
Equipment	16,800	16,800	67,993	84,793
Indirect cost	24,559	28,961	74,046	103,007
Totals	79,973	84,044	307,233	391,277

4. References

[1] K. Birman and T. Joseph. Communication Support for Fault Tolerant Process Groups. Proc *Asilomar workshop on fault-tolerant distributed computing*, Springer Verlag, March 1986.

[2] Cheriton, D., Zwaenepoel, W. Distributed process groups in the V kernel. *ACM TOCS* **3**, 2 (May 1985), 77-107.

[3] Cooper, E. Replicated distributed programs. *Proc. ACM 10th SOSP*, Orcas Island, WA (Dec. 1985), 63-78.

[4] K. Birman and T. Joseph. Reliable communication in the presence of failures. Department of Comp. Sci., Cornell University, Revision of TR 85-694 (Aug. 1985; revised Aug. 1986). Accepted for publication, *ACM TOCS*.

[5] K. Birman, T. Joseph, P. Stephenson. Programming with fault-tolerant bulletin boards. Department of Comp. Sci., Cornell University, *forthcoming* (August 1986).

[6] T. Raeuchle. Concurrency control for nested objects. Dept. of Computer Science, Cornell Univ. *Ph.D. dissertation* Forthcoming (Aug. 1986).